**1)Расскажите, как генерируется исключение.**

Обычно система сама генерирует исключения при определенных ситуациях, например, при делении числа на ноль. Но язык C# также позволяет генерировать исключения вручную с помощью оператора **throw**. То есть с помощью этого оператора мы сами можем создать исключение и вызвать его в процессе выполнения.

**2)Расскажите методику обработки исключений.**

Короче, когда оно генерируется, система ищет ближайший подходящий блок catch. После этого обрабатывается блок finally (если он есть). Потом либо конец выполнения программы, либо, если это указано в блоке finally или catch, переходим на другой блок кода

**3)Какое ключевое слово служит для обозначения блока кода, в котором можно генерировать исключение?**

Try { }

**4) Какие ключевые слова используются для обработки и генерации исключений? Расскажите об механизме обработке исключения?**

Exception, try, catch, finally

Механизм

**1) Исключение не произошло**

1.1. try выполняем до конца

1.2. catch пропускаем

1.3. finally выполняем

**2) Исключение произошло**

2.1. выполнение try прекращается (все что идет

за возникшим исключение игнорируется)

2.2 ищем блок catch на соответствие по типу

исключения

2.2.1. если нет catch

2.2.1.1 разматывает стек, локальные объекты,

выходят из области видимости

2.2.1.2 снова генерируется исключение в точке

вызова метода

2.2.1.3. если блока не найдено, то сообщение -

необработанное исключении-> дальнейшее

выполнение программы останавливается

2.2.2 сatch найден

2.2.2.1 Передается управление ближайшему

**catch-обработчику, совместимому с** типом

выброшенного исключения

2.2.2.2. объект-исключения передается, если это

предусмотрено, обработчику в качестве

параметра.

2.3. переходим в finally

2.3.1. если нет finally

2.3.1.1. выполнение программы продолжается

начиная с позиции, след. за последним

обработчиком данного блока try

**5) Что будет, если в программе нет предложения catch, способного обработать исключение?**

Дальнейшее выполнение программы останавливается.

**6) Что такое фильтры исключения? Приведите пример**

Фильтр исключения позволяет указать дополнительные условия, при

которых используется обработчик исключения.

Эти условия принимают форму булева выражения, перед которым

ставится ключевое слово when.

int x = 1;

int y = 0;

try

{

    int result = x / y;

}

catch(DivideByZeroException) when (y==0 && x == 0)

{

    Console.WriteLine("y не должен быть равен 0");

}

catch(DivideByZeroException ex)

{

    Console.WriteLine(ex.Message);

}

**7) Могут ли исключения быть вложенными?**

Могут

**8) Какой синтаксис нужно использовать в C# для отлова любого возможного исключения?**

Try{}

catch без параметров либо catch (Exception)

**9) Чем следует руководствоваться при размещении обработчиков исключения?**

Их иерархией, исключения, которые наследуются от других исключений, должны быть первыми, в самом конце следует ставить catch (Exception)

**10) Что будет выведено на консоль в результате выполнения фрагмента листинга?**

static void Main(string[] args)

{

string[] str = new string[5];

try

{

str[4] = "anything";

Console.WriteLine("It's OK");

}

catch (IndexOutOfRangeException e)

{

Console.WriteLine("IndexOutOfRangeException");

}

catch (Exception e)

{

Console.WriteLine("Exception");

}

}

**}**

Исключение не сгенерируется

**11) Как повторно сгенерировать то же самое исключение в блоке обработчике catch?**

Throw;

**12) Какие методы содержаться в классе Exception? Где и как их можно использовать?**

Свойство Message содержит символьную строку, описывающую характер ошибки;

Свойство StackTrace— строку с вызовами стека, приведшими к исключительной ситуации

ТагgetSite получает объект, обозначающий метод, сгенерировавший исключение.

Кроме того, в классе Exception определяется ряд методов. Чаще всего приходится пользоваться методом ToString()

Макрос проверяет переданное ему выражение на истинность (то есть *true* или *false*). Если выражение *true* — программа выполняется дальше, если *false* — программа аварийно завершается

После первого выполнения выражение, переданное макросу, равнялось *true*, поэтому программа продолжила свою работу. Второй вызов *assert* получил в качестве результата выражения *false*. В этот момент, в стандартный поток ошибок(*stderr*) была записана информация о вызове, и вызвана функция *abort()*.